(54) **System for compressing and decompressing binary representations of dithered images**

(57)   A method for compression and decompression of dithered images is disclosed. Logical units (tiles) of the binary representation are classified (205) into equivalence classes which are then compressed (206). Each equivalence class represents tiles having similar gray levels (i.e. the same number of black pixels), but which may have different sequences of black and white pixels. Each equivalence class has associated with it a predefined set of rendering exemplars. Each of the exemplars has a similar gray level. Upon decompression, each instance of an equivalence class takes on the value of one of the rendering exemplars which is selected pseudorandomly. This effectively causes the image to be redithered so there is no loss of critical image information on decompression.
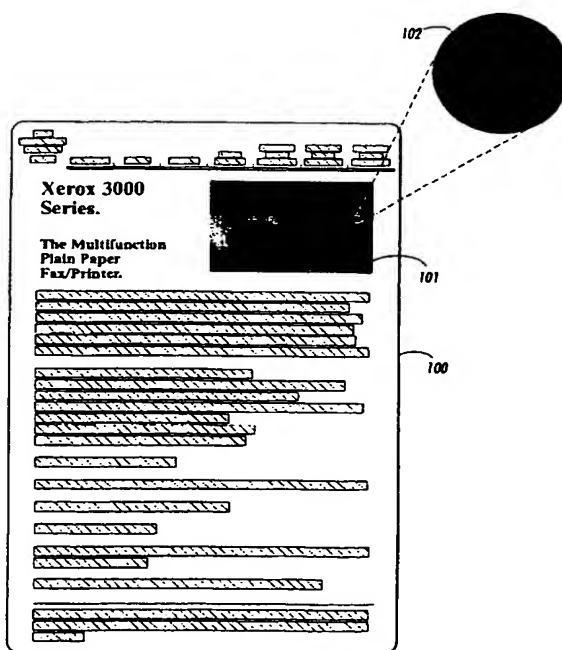
FIG. 1

Description

[0001] The present invention is related to the field of data compression, and in particular to LOSSY compression of dithered images.

[0002] A major stumbling block to common use of digitized images is their size. An 8.5 x 11 image at 300 dots per inch (dpi) contains roughly 8,000,000 pixels. Even after binarization of a scanned image reduces the number of bits per pixel to 1, this is still 1 megabyte. Compression techniques are typically characterized as LOSSLESS or LOSSY. In LOSSLESS compression, no data is lost in the compression and subsequent decompression. In LOSSY compression, a certain amount of data is lost but it is acceptable since the essence of the compressed data is retained after decompression.

[0003] Common LOSSLESS compression techniques for binary images, like CCITT Group 3 or Group 4 or MMR can compress a binary image by factors of 10 to 20. This is still large when compared to the synthetic electronic form used to create a comparable image. Moreover, such compression techniques do not perform well for dithered images. This is because such compression techniques generally depend on the compressor's ability to predict the value of a pixel given neighboring pixels. Dithered images contain many very tiny dots which are intentionally arranged in a pseudo-random pattern. In these images, it is quite difficult to predict the value of a pixel, thus such compression techniques perform poorly.

[0004] Vector Quantization is another LOSSY based method for image compression that is well known. A vector quantizer (VQ) is a quantizer that maps k-dimensional input vectors into one of a finite set of k-dimensional reproduction vectors, or codewords. For image compression the input vector is a fixed grouping of pixels. A VQ can be divided into two parts: an encoder and a decoder. The encoder maps the input vector into a binary code representing the index of the selected reproduction vector, and the decoder maps the binary code into the selected reproduction vector. The reproduction vector becomes the decompressed value of the input vector.

[0005] Typically the decoder operates using a simple lookup table. To obtain acceptable results upon decompression, the number of reproduction vectors, and resulting lookup table, can be quite large. As the lookup table may be part of the compressed data stream, a large lookup table is undesirable.

[0006] Vector quantization is conceptually similar to a known method for performing compression on text images by grouping the symbols found into equivalence classes. In this method, symbols are extracted from the binary image and matched to templates for one or more equivalence classes. In order to get good compression, a classifier should operate with a small number of classes.

[0007] An example of image compression based on symbol matching is described in co-pending U.S. Patent Application Serial No. 08/575,305 filed December 20, 1995, entitled "Classification Of Scanned Symbols Into Equivalence Classes". A further example of image compression based on symbol matching is described in U. S. Patent No. 5,303,313 entitled "Method and Apparatus For Compression Of Images", Mark et al., issued April 12, 1994 (the '313 patent). In the '313 patent an image is "precompressed" prior to symbol matching. The '313 patent describes using run-length encoding for such precompression. Symbols are extracted from the run-length representation. A voting scheme is used in conjunction with a plurality of similarity tests to improve symbol matching accuracy. The '313 patent further discloses a template composition scheme wherein the template may be modified based on symbol matches.

[0008] However, the aforementioned symbol based compression techniques do not compress particularly well with respect to pictorial images, in particular dithered images. This is because the pseudo random patterns typically causes a high number of equivalence classes to be created and because a very large number of symbols must be classified (often each dot would be interpreted as a separate symbol).

[0009] A system for compressing and decompressing binary representations of dithered images is disclosed. The currently preferred embodiment of the present invention provides a LOSSY method for compressing dithered images. In LOSSY compression, some of the original image data is lost. It has been determined that for dithered images such as halftoned and error-diffused images an exact reproduction of the original image may not be necessary for acceptable results. The present invention incorporates the idea that maintaining the exact position of edges in a dithered image is not as important as maintaining the gray levels. This is because dithered images contain dots which are intentionally arranged in a pseudo random pattern. Through arrangement of pixels in such a pseudo random pattern, undesirable artifacts such as streaks or lines, are avoided.

[0010] The compression method of the present invention is generally comprised of the steps of defining a plurality of equivalence classes for tiles of multi-pixel binary encoded data contained in said binary encoded image, wherein tiles are of a first predetermined organization of binary data and each equivalence class has defined and associated therewith one or more rendering exemplars; classifying each tile in said binary encoded image into an equivalence class and encoding the equivalence classes by scanline into sequences of literal elements and copy elements, wherein literal elements direct decompression to find the equivalence class in the compressed data stream and copy elements direct decompression to find the equivalence class in the immediately preceding decompressed scanline. On decompression, the sequence of literal elements and copy elements are decoded into their respective equivalence classes a scanline at a time and then a corresponding rendering

exemplar is selected for each equivalence class.

Figure 1 is an illustration of a dithered image which may be compressed using the LOSSY compression method of the currently preferred embodiment of the present invention;

Figure 2 is a flowchart describing the general steps for the data compression and decompression method of the currently preferred embodiment of the present invention;

Figure 3 is a table containing the values of the encoded equivalence classes as may be utilized in the currently preferred embodiment of the present invention;

Figure 4 is a flowchart illustrating the steps for compressing a scanline using the data compression method of the currently preferred embodiment of the present invention;

Figure 5 is a table used for identifying which equivalence class a tile belongs to in the currently preferred embodiment of the present invention;

Figure 6 is a block diagram showing the functional components of a compression system for practicing the compression method of Figure 4;

Figure 7 is a block diagram illustrating a compressed scanline data stream as may be utilized in the currently preferred embodiment of the present invention;

Figure 8 is a flowchart illustrating the steps for decompressing a compressed data stream using the decompression method of the currently preferred embodiment of the present invention;

Figure 9 is a block diagram showing the functional components of a decompression system for practicing the compression method of Figure 8;

Figure 10 is a table containing the values of the rendering exemplars of the currently preferred embodiment of the present invention; and,

Figure 11 is an illustration of a computer based system upon which the currently preferred embodiment of the present invention may be utilized.

[0011] A system for compressing and decompressing binary representations of continuous tone images is disclosed. The present invention may be used in various applications requiring or benefiting from data compression. Such applications may be found as part of an overall image processing system or as stand-alone applications. The currently preferred embodiment of the present invention is implemented as software running on a computer based system. The software is written in the C programming language. The present invention has been preferably implemented for compression of pictorial image data.

[0012] The following terms take on the accompanying meaning in this description:

[0013] **Image** refers to the markings on or appearance of a medium.

[0014] **Image data** refers to a representation of an image which may be used for recreating the image.

[0015] **Pictorial image** refers to non-textual and non line art markings on a medium.

[0016] **Tile or Tiles** refers to a logical organization of pixels as the primitive object which is classified.

[0017] An **equivalence class** is a set of tiles found in an image that can be substituted for one another without changing the appearance of an image in an objectionable way.

[0018] The **rendering exemplars** of the equivalence class are the set of pixel configurations for the equivalence class, one or more of which will be substituted for a member of the equivalence class when the image is decompressed or otherwise recreated. The collection of rendering exemplars for the equivalence classes are referred to as a **rendering dictionary.**

[0019] **Copy element** refers to a data element in a compressed data stream which instructs decompression to obtain equivalence class values from the preceding decoded scanline.

[0020] **Literal element** refers to a data element in a compressed data stream which instructs decompression to obtain equivalence class values in the element itself.

[0021] **Compressed Data Stream** refers to a compressed representation of an image comprised of copy and literal elements and possibly a corresponding rendering dictionary.

[0022] The currently preferred embodiment of the present invention provides a LOSSY method for compressing dithered images. In LOSSY compression, a certain amount of the data is altered during compression and subsequent decompression. It has been determined that for both halftoned and error-diffused images an exact reproduction of a original image is not necessary for acceptable results. The present invention incorporates the idea that maintaining the exact position of edges in a dithered image is not as important as maintaining the gray levels. This is because dithered images contain dots which are intentionally arranged in a pseudo random pattern. Through arrangement in such a pseudo random pattern, undesirable artifacts such as streaks or lines, are avoided.

[0023] A dithered pictorial image which may be compressed using the method of present invention is illustrated in Figure 1. Referring to Figure 1, a document image 100 has both text and a pictorial areas. The pictorial area 101 is indicated and shown in a magnified view 102. As described above, the magnified view 102 shows the pictorial area 101 being comprised of dots in a random pattern.

[0024] It should be noted that the text area may also be compressed using the method of the present invention, but the resulting decompressed image may lose too much information (i.e. it will look bad). Various techniques are known to separate text and pictorial areas from a scanned image and applying different compres-

sion schemes on each. The method of the present invention may operate in such an implementation.

[0025] The present invention is accomplished by first defining a set of equivalence classes for a logical unit, i.e. a tile, of the image according to a set of predetermined rules. When compressing, each tile in the image is assigned to an equivalence class. Each tile in the equivalence class will have the same gray level, but may have different sequences of black and white pixels.

[0026] Further associated with each equivalence class is a set of rendering exemplars. Each rendering exemplar will produce the same gray level (i.e. they have the same number of black pixels). Upon decompression, the set of equivalence classes representing the image are decoded. For each equivalence class a rendering exemplar is selected from the corresponding set based on some pseudo random criteria (e.g. the scanline).

[0027] Another way to characterize the present invention is view it as vector quantization with a reduced size rendering dictionary and wherein the pictorial image is redithered on decompression.

[0028] Figure 2 is a flowchart which describes the general steps of the method of the present invention. First, a set of equivalence classes and corresponding rendering dictionary are created for a particular tile configuration, step 201. In order for a document image to be compressed, a document is scanned to create image data, step 202. The image data is typically a bit-mapped representation of the image. The pictorial portions of the image data are segmented from the text portions of the image data, step 203. It is the pictorial portion that is processed by the present invention. It should be noted that preferably any text contained in the pictorial portion also be segmented out. Further, the entire image may be comprised of pictorial data. The pictorial portion of the image data is then converted into a binary representation to create a representation where each pixel is represented by a single bit, step 204. A multi-level image may typically have a pixel represented by a multi-bit data operand. In order to convert it into a single bit value, a dithering operation is typically performed. This dithering operation provides a smooth and visually pleasing transition between different areas of the image.

[0029] Each of the tiles in the binary representation are then extracted and categorized into a particular equivalence class, step 205. An equivalence class identifier is provided for each tile. In the currently preferred embodiment, a tile is a 1 pixel high and 8 pixel wide block. It is assumed that the equivalence classes are predefined in a manner which corresponds to the tile size. The stream of equivalence class identifiers are then encoded, step 206. The steps 205 and 206 as described herein are in general terms. The preferred manner in which steps 205 and 206 are implemented is described in greater detail below. However, it should be noted that other known techniques may be utilized and which would not cause departure from the spirit and

scope of the present invention. In any event, the encoding step 206 results in a compressed image data stream. The compressed data stream is comprised of a sequence of copy elements (for instructing a decompression process to find each equivalence class from the corresponding position(s) in the preceding decoded scanline) and literal elements (for instructing the decompression process to derive each equivalence class directly from the compressed data stream). Each copy and literal element will represent some number of tiles extracted from the image data. Depending on the implementation of the present invention, the compressed image data stream may or may not include the rendering dictionary.

[0030] It should be noted that in the currently preferred embodiment the encoding of the equivalence class representations is made immediately after the tiles of a scanline have been converted into equivalence classes. This may in fact minimize the amount of internal memory required for performing the compression (e.g. by limiting it to two scanline buffers). However, it would be possible to perform the encoding after all the equivalence classes have been identified for the image, which may enable the use of alternative encoding schemes.

[0031] The compressed data stream may then be stored or transmitted, depending on the purpose for the compressed data stream, step 207.

[0032] When the compressed data stream is to be decompressed, the equivalence class identifier encoding is decoded in order to obtain the equivalence class representation, step 208. A binary representation of the image is created using the equivalence class representations, the rendering dictionary and some pseudo random input such as the number of the current scanline, step 209. Note that this binary representation would typically be different from the originally created binary representation in step 204 since for any particular tile, the specific rendering exemplar used may have a different pixel configuration of the original tile. This in effect causes the image to be redithered.

[0033] In the currently preferred embodiment, the image is broken into tiles 1 pixel tall and 8 pixels wide. Utilization of other tile sizes and dimensions are possible and would not depart from the spirit and scope of the present invention. The contents of each tile is then classified into one of a plurality of predefined equivalence classes. The equivalence classes are defined so that upon decompression, re-dithering may occur.

[0034] In order to achieve significant compression, it is desirable to minimize the number of equivalence classes. For a tile size of 1 X 8 binary values, there could theoretically be a maximum of 256 equivalence classes. An efficient method of defining a minimal number of equivalence classes is needed. In the currently preferred embodiment there are 47 predefined equivalence classes. This number was experimentally determined to provide acceptable visual results on decompression. However, utilization of a different number of equivalence

classes is possible and would clearly be within the spirit and scope of the present invention.

[0035]   The general rules used for creating the equivalence classes are as follows:

1. All members of an equivalence class should have the same number of black pixels.

2. An isolated black pixel can move anywhere inside a tile and still be in the same equivalence class.

3. An isolated block of 2 black pixels can move left or right two pixel positions within a tile while remaining in the same equivalence class.

4. An isolated block of 3 or 4 black pixels can move left or right by 1 pixel position within a tile while remaining in the same equivalence class.

5. A block of 2 or more black pixels attached to the left or right edge of a tile must stay attached to that edge.

6. If there are more black pixels than white pixels in a tile, all the above rules are applied to groups of isolated white pixels instead of applying them to groups of isolated black pixels. For example, this means that the equivalence classes that contain 5 . black pixels can be derived from the equivalence classes that contain 3 black pixels by simply inverting all the pixels in each tile of each class.

[0036]   The equivalence classes of the currently preferred embodiment are illustrated in the table of Figure 3. In reviewing the table of Figure 3 it should be noted that the tile configurations are illustrated in their hex rather than the binary values of the actual tile. The hex/ binary equivalents are: 0=0000, 1=0001, 2=0010, 3=0010, 4=0100, 5=0101, 6=0110, 7=0111, 8=1000, 9=1001, A=1010, B=1011, C=1100, D=1101, E=1110 and F=1111. It should be noted that in Figure 3, the classes 0-7 may appear to be out of order. It has been determined that the ordering of classes in this manner may possibly lead to greater compression since there are many transitions amongst this group of classes and special coding may be utilized. However, utilization of different ordering of the classes would be within the spirit and scope of the present invention.

[0037]   In the currently preferred embodiment, images are converted to their equivalence class representation and then encoded a scanline at a time. Figure 4 is a flowchart illustrating the steps for compressing a scanline in the currently preferred embodiment of the present invention and correspond to steps 205-206 of Figure 2. Referring to Figure 4, a tile in a first scanline is extracted from the image, step 401. This extraction is merely taking a grouping of 8 pixels, or as they are represented by binary data, a byte of data in the scanline. The equivalence class in which the tile belongs is identified, step 402. The currently preferred embodiment uses the table in Figure 5 to map a tile to its equivalence class. Referring to Figure 5, the column entry represents the 4 left most bits and the row entry represents the 4 right most

bits of a tile. Equivalence class identification is then merely a table-look up.

[0038]   The equivalence class information is then stored in a first scanline buffer, step 403. As will become apparent in the description below, the first scanline buffer will become a "reference" buffer for encoding the second scanline. The steps 401-403 are repeated until all the tiles in the first scanline are converted into their respective equivalence classes, step 404. The equivalence classes of the first scanline are then encoded into a literal element, step 405. In this case it will be an encoding where the length is the length of the scanline followed by an encoding by equivalence class identifiers. It should also be noted that for the first scanline, the first element is a copy element having a zero length. Further, in the currently preferred embodiment a Huffman encoding of literal elements and lengths is performed. The Huffman codes utilized are based on experimentation and observation of the frequency patterns of equivalence classes. Utilization of Huffman codes in this manner is known in the art.

[0039]   A second scanline is then converted into a set of equivalence classes using basically the same steps as described in steps 401-403 except that the equivalence classes are stored in a second scanline buffer, step 406. Encoding of the equivalence classes henceforth occurs differently since there is now a previous encoded scanline to compare to. It should be noted that encoding for the subsequent scanlines always results in a sequence of alternating copy and literal elements. First a tile's equivalence class of the second scanline buffer is compared to the corresponding position in the first scanline buffer, step 407. If there is a match, the successive equivalence class IDs are compared from the respective buffers and a match length calculated until there is no longer a match, step 408. A copy elements match length is then encoded, step 409. Copy elements are also Huffman coded.

[0040]   If there was no match in step 407, a zero match length is encoded in the copy element, step 410. Next, the successive equivalence class definitions in the second scanline buffer are compared to determine if they are in the same equivalence class and a length calculated until a matching pair of equivalence classes is encountered, step 411. A length for a literal element is then encoded along with the equivalence class identifiers represented by the literal element, step 412. The alternating creation of copy elements and literal elements is then repeated for the remainder of the scanline, step 413.

[0041]   For the next (e.g. third) scanline, the second scanline buffer is used as the "reference" buffer, while the first scanline buffer is used as the buffer into which equivalence classes are written. This switching of the use of the first and second scanline buffers as the "reference" buffer continues for the entire image.

[0042]   It is noted that this technique of using a previously encoded scanline as a reference buffer is a well

known technique used for example in CCITT Group 4 compression. However, other types of encodings for a stream of equivalence class identifiers could be utilized such as a lempel-ziv technique, without departing from the spirit and scope of the present invention. Figure 6 is a block diagram illustrating the functional components for performing the compression method of Figure 4. Referring to Figure 6, an Uncompressed Image Data Stream 601 is coupled to an Equivalence class identifier 602. The Uncompressed Image Data Stream 601 contains the original uncompressed scanline data. Further coupled to the equivalence class identifier 602 is an equivalence class map 603. The equivalence class map 603 may be stored in a conventional storage device and will contain the information as shown in Figure 5. The output of the equivalence class identifier 602 switches between scanline buffers 604 and 605. As noted above, since encoding of the equivalence classes depends on the equivalence classes of the previous scanline, a copy of the previous scanline is maintained. Each of the scanline buffers 604 and 605 is further coupled to equivalence class encoder 606. The equivalence class encoder 606 compares the contents of the scanline buffers and encodes them into copy or literal elements as appropriate which are stored as a Compressed Data Stream 607. Further coupled to the equivalence class encoder is encoding tables 608. The encoding tables 608 contain the information needed to perform the Huffman encoding on the copy and literal elements.

[0043] Figure 7 is a block diagram illustrating a compressed scanline data stream. Referring to Figure 7, as noted above for a compressed scanline the first element is always a copy element, here copy element 701. The information contained in the copy element 701 is a match length. The next element, if one exists, will be a literal element 702. Note that no literal element would be encoded if the equivalence class identifiers for the current scanline and the reference scanline were identical. The literal element contains a length along with a corresponding number of equivalence class identifiers 703. If there is a next element it will be a copy element 704, again comprising a match length as the primary information. If the scanline is not completed, this will again be followed by a literal element 705 and equivalence class identifiers 706. In practice, a copy element is merely an encoding of the match count while a literal element is an encoding of a length followed by the corresponding number of equivalence class encodings.

[0044] As noted above compression is further achieved through a Huffman encoding of the length information for the literal and copy elements of the created compressed data stream. It should be noted that the codes for the lengths of literal and copy elements are different. This is because the observed frequency characteristics for the two types of elements differ.

[0045] Further, the currently preferred embodiment contains other special encodings for the lengths for copy elements. A code is provided for indicating copying to the end of scanline. Other codes are provided to indicate copying to a next (or next after the next), key point. A key point is defined as a point in the previously encoded scanline where a transition was made from encoding of a copy element to encoding of a literal element. Note that this type of encoding is also done in CCITT Group 4 encoding, so no further description of this type of encoding is deemed necessary.

[0046] As described above, each equivalence class is represented by a plurality of rendering exemplars. In the currently preferred embodiment, each equivalence class is represented by a set of 8 rendering exemplars. Note that in some instances the 8 rendering exemplars may not all be unique (e.g. the case where an equivalence class only represents only a single tile configuration). However, for the most part, the 8 rendering exemplars were chosen because they present a pleasing visual appearance when used to re-dither the equivalence class on decoding.

[0047] As described above, in the currently preferred embodiment, scanline encoding/compression is performed based on the content of the immediately previous scanline. A similar process is performed in decoding/decompression. Figure 8 is a flowchart describing the steps for decompression. Referring to Figure 8, a first scanline is decoded to create a decoded scanline in a current scanline buffer, step 801. The decoded scanline is comprised of a scanline of equivalence class identifiers. In the currently preferred embodiment, information indicating the length of a scanline is obtained from descriptive information which describes the image. For example, a document image encoded in the Tagged Information File Format (TIFF) will include a tag having image width information from which scanline information may be derived. So the decoding occurs until the determined number of equivalence class identifiers have been decoded. Rendering exemplars for each of the equivalence classes are then identified and output as a decompressed scanline, step 802. In any event, once a scanline is decoded and equivalence class exemplars identified, the "current" scanline buffer becomes a "reference scanline buffer" and the next scanline becomes the current scanline and is decoded based in part on the decoded equivalence class identifiers found in the reference scanline buffer. The process begins by first obtaining a copy element for the current scanline and switching the current scanline buffer to a reference scanline buffer, step 803. As described above, in the currently preferred embodiment, this first element will always be a copy element, or more precisely it is presumed to be a count (which may be zero). The copy element is decoded to obtain a length M, step 804. It should be noted that as described above, the length M may represent a key point in the reference scanline. In any event, the number of equivalence class IDs represented by Length M in the corresponding positions in the reference scanline buffer are then copied to the current scanline buffer, step 805. Note that if length M is

zero, then no equivalence class IDs are copied.

[0048] The next literal element, if one exists, is decoded to determine a length N, step 806. The N equivalence class IDs included in the literal element are then copied into the current scanline buffer, step 807. The decoding of the current scanline continues per steps 803-807 continues until enough equivalence class IDs have been decoded to fill the scanline, step 808. When the decoding of the current scanline is completed, the rendering exemplars for the various equivalence classes in the current scanline are obtained and output as a decoded scanline, step 809.

[0049] The steps 803-809 are then repeated for the remainder of the scanlines in the image, step 810.

[0050] Figure 9 is a block diagram of the functional components of a decompression system for performing the decompression described in the flowchart of Figure 8. Compressed Data stream 901 is input to an element decoder unit 902. The element decoder unit 902 is comprised of a copy element decoder 903 and a literal element decoder 904. The copy element decoder 903 will decode the match length information for copy elements. The literal element decoder 904 will decode the count information as well the corresponding number of equivalence class IDs. Each of the copy element decoder 903 and the literal element decoder 904 is coupled to a buffer unit 905. The buffer unit 905 is comprised of scanline buffer 1 906 and scanline buffer 2 907. The buffer unit 905 further contains buffer control 908 which manages access to the buffers (e.g. determines which is the reference buffer used by copy element decoder, when the buffer can be output, etc..)

[0051] The copy element decoder 903 will send control information to the buffer unit 905 for controlling copying of equivalence class IDs between scanline buffer 1 906 and scanline buffer 2 907. The literal element 904 will send decoded equivalence class IDs to the buffer unit 905 for storing in one of scanline buffer 1 906 or scanline buffer 2 907.

[0052] Further coupled to the buffer unit 905 to receive scanlines of decoded equivalence class IDs is dither unit 909. The dither unit 909 is used for determining the rendering exemplar to be used for rendering the corresponding equivalence class. Coupled to the dither unit 909 is a rendering dictionary 910. The rendering dictionary contains the sets of rendering exemplars used for each equivalence class. The output of the dither unit 909 is the decompressed image data stream 911.

[0053] Figure 10 is a table illustrating the rendering exemplars for the equivalence classes of the currently preferred embodiment of the present invention. The table of Figure 10 may also be considered the rendering dictionary for the equivalence classes and would be contained in the rendering dictionary 910 of Figure 9. The rendering dictionary enables the decompressed pictorial image to remain dithered (or to be re-dithered). The creation of the rendering exemplars is based in part on the number of tile configurations and an estimate of the aesthetic effect obtained by filling a region with a single equivalence class so as to create no streaks or bars. Note that other sets of rendering exemplars may be used, so long as an acceptable aesthetic effect is maintained, without departing from the spirit and scope of the present invention.

[0054] In the currently preferred embodiment, the rendering exemplar used is determined by the selecting the exemplar corresponding to the value of the scanline number being processed modulo the number of exemplars, e.g. 8. So for example, for the 25th scanline, 25 modulo 8, i.e. 1, would cause the second exemplar in the corresponding set to be used. Note that the sets are numbered 0-7. It should be noted that other methods could be used for selecting the exemplar (e.g. a completely random scheme), and would not cause departure from the spirit and scope of the present invention.

[0055] The computer based system on which the currently preferred embodiment of the present invention may be used is described with reference to Figure 11. Referring to Figure 11, the computer based system is comprised of a plurality of components coupled via a bus 1101. The bus 1101 illustrated here is simplified in order not to obscure the present invention. The bus 1101 may consist of a plurality of parallel buses (e.g. address, data and status buses) as well as a hierarchy of buses (e.g. a processor bus, a local bus and an I/O bus). In any event, the computer system is further comprised of a processor 1102 for executing instructions provided via bus 1101 from Internal memory 1103 (note that the Internal memory 1103 is typically a combination of Random Access or Read Only Memories). Such instructions are those that are preferably implemented in software for carrying out the processing steps outlined above in the flowcharts of Figures 2, 4 and 8. The processor 1102 and Internal memory 1103 may be discrete components or a single integrated device such as an Application Specification Integrated Circuit (ASIC) chip. Further the combination of processor 1102 and Internal Memory 1103 comprise circuitry for performing the functionality of the present invention so that the currently preferred embodiment of the present invention could be implemented on a single ASIC or other integrated circuit chip.

[0056] Also coupled to the bus 1101 are a keyboard 1104 for entering alphanumeric input, external storage 1105 for storing data such as a compressed text image data file, a cursor control device 1106 for manipulating a cursor, and a display 1107 for displaying visual output. The keyboard 1104 would typically be a standard QWERTY keyboard but may also be a telephone like keypad. The external storage 1105 may be fixed or removable magnetic or optical disk drive. The cursor control device 1106 will typically have a button or switch associated with it to which the performance of certain functions can be programmed. Further coupled to the bus 1101 is a scanner 1108. The scanner 1108 provides a means for creating a bitmapped representation of a medium (i.e. a scanned document image).

[0057] Further elements that could typically be coupled to the bus 1101 would include printer 1109, facsimile element 1110 and network connection 1111. The printer 1109 could be used to print the bitmapped representation. The facsimile element 1110 may contain an element used to transmit a image data that has been compressed using the present invention. Alternatively, the facsimile element 1110 could include an element for decompression of a document image compressed using the present invention. The network connection 1111 would be used to receive and/or transmit data containing image data. Thus, the image data utilized by the present invention may be obtained through a scanning process, via a received fax or over a network.

Claims

1. A method for processing a binary representation of a dithered image so that it may be compressed without losing essential information, said method comprising the steps of:

   a) defining a plurality of equivalence classes for tiles of multi-pixel binary data contained in said dithered image, wherein tiles are of a predetermined organization of binary data, each of said equivalence classes further having defined and associated therewith a plurality of rendering exemplars having the same gray level;
   b) identifying one equivalence class for each tile in said dithered image;
   c) encoding each of said equivalence classes into an alternating sequence of literal elements and copy elements, said literal elements containing equivalence class information for one or more consecutive tiles and said copy elements containing information for copying previously encoded equivalence class information.

2. A method for decompressing a compressed binary representation of a dithered image, said dithered image organized into a plurality of tiles representing multiple pixels, said method comprising the steps of:

   a) decoding said compressed binary representation into a plurality of literal elements and copy elements;
   b) decoding each of said literal elements and copy elements into equivalence class identifiers, each equivalence class identifier corresponding to a tile of said dithered image;
   c) selecting an equivalence class exemplar from a rendering dictionary for each equivalence class identifier, said rendering dictionary comprised of a plurality of representative tile configurations for said class.

3. A system for compression and decompression of dithered images, said system comprising:

   input means for receiving a dithered image;
   a processor for performing operations for compressing said dithered image and decompressing a compressed representation of said dithered image;
   storage means for storing tile equivalence class definitions, each of said tile equivalence class definitions for categorizing a plurality of tile configurations into a single class;
   said storage means further for storing data including operations for compressing said dithered image, said operations including:
   operations for creating a binary representation of said dithered image;
   operations for extracting tiles from said binary representation and determining the equivalence class for a tile;
   operations for compressing a predetermined collection of tiles based on their determined equivalence class creating an alternating sequence of literal elements and copy elements;
   said storage means further for storing equivalence class exemplars, each of said equivalence class exemplars comprising a plurality of representative tile configurations for said class, each of said tile configurations have the same gray level;
   said storage means further for storing data including operations for decompressing a compressed representation of a dithered image including:
   operations for decompressing sequences of literal and copy elements into instances of equivalence classes;
   operations for selecting one of said plurality of representative tile configurations for an instance of an equivalence class.

Xerox 3000 Series.

The Multifunction Plain Paper Fax/Printer.

FIG. 1

CREATE A SET OF EQUIVALENCE
CLASSES WITH CORRESPONDING
RENDERING DICTIONARY
— 201

SCAN DOCUMENT PAGE
TO CREATE IMAGE DATA
— 202

SEGMENT IMAGE DATA TO
SEPARATE PICTORIAL PORTIONS
FROM TEXT PORTIONS
— 203

CONVERT PICTORIAL PORTION TO
BINARY REPRESENTATION
— 204

CATEGORIZE TILES OF BINARY
REPRESENTATION INTO EQUIVALENCE CLASSES
— 205

ENCODE EQUIVALENCE CLASSES
INTO COMPRESSED IMAGE DATA STREAM
— 206

STORE, TRANSMIT OR OTHERWISE
PROCESS COMPRESS IMAGE DATA STREAM
— 207

DECODE COMPRESSED IMAGE DATA
STREAM INTO EQUIVALENCE CLASSES
— 208

RECREATE BINARY REPRESENTATION
USING RENDERING DICTIONARY AND
PSEUDO-RANDOM INPUT
— 209

FIG. 2

CLASS NUMBER	CLASS DESCRIPTION	TILE CONFIGURATIONS IN CLASS
0	3 Isolated White Pixels	57, 5B, 5D, 5E, 6B, 6D, 6E, 75, 76, 7A, AB, AD, AE, B5, B6, BA, D5, D6, DA, EA
1	2 Isolated White Pixels	5F, 64, 77, 7B, 7D, 7E, AF, B7, BB, BD, BE, D7, DB, DD, DE, EB, ED, EE, F5, F6, FA
2	1 White Pixel	7F, BF, DF, EF, F7, FB, FD, FE
3	8 Black Pixels	FF
4	8 White Pixels	00
5	1 Black Pixel	80, 40, 20, 10, 08, 04, 04, 01
6	2 Isolated Black Pixels	A0, 90, 88, 84, 82, 81, 50, 48, 44, 42, 41, 28, 24, 22, 21, 14, 12, 11, 01, 09, 05
7	3 Isolated Black Pixels	A8, A4, A2, A1, 94, 92, 91, A1, 94, 92, 91, 8A, 89, 85, 54, 52, 51, 4A, 49, 45, 2A, 29, 25, 15
8	2 Black Pixels	C0
9	2 Black Pixels	60, 30, 18
10	2 Black Pixels	0C, 06
11	2 Black Pixels	03
12	3 Black Pixels	E0
13	3 Black Pixels	70, 38
14	3 Black Pixels	1C, 0E
15	3 Black Pixels	07
16	3 Black Pixels	D0, C8, C4, C2, C1
17	3 Black Pixels	68, 64, 62, 61, B0, 34, 32, 31, 98, 58, 1A, 19
18	3 Black Pixels	8C, 4C, 2C, 0D, 86, 46, 26, 16
19	3 Black Pixels	83, 43, 23, 13, 0B
20	4 Black Pixels	F0
21	4 Black Pixels	78, 3C
22	4 Black Pixels	1E, 0F
23	4 Black Pixels	E8, E4, E2, E1
24	4 Black Pixels	74, 72, 71, B8, 3A, 39
25	4 Black Pixels	9C, 5C, 1D, 8E, 4E, 2E
26	4 Black Pixels	87, 47, 27, 17
27	4 Black Pixels	D4, D2, D1, CA, C9, C5
28	4 Black Pixels	6A, 69, 65, B4, B2, B1, 35, 9A, 99, 5A, 59
29	4 Black Pixels	AC, 8D, 4D, 2D, A6, 96, 56
30	4 Black Pixels	A3, 93, 8B, 53, 4B, 2B
31	4 Black Pixels	D8, CC, C6, 63, 33, 1B
32	4 Black Pixels	6C, 66, 36
33	4 Black Pixels	C3
34	4 Black Pixels	AA, A9, A5, 95, 55
35	5 Black Pixels	1F
36	5 Black Pixels	8F, C7
37	5 Black Pixels	E3, F1
38	5 Black Pixels	F8
39	5 Black Pixels	2F, 37, 3B, 3D, 3E
40	5 Black Pixels	97, 9B, 9D, 9E, 4F, CB, CD, CE, 67, A7, E5, E6
41	5 Black Pixels	73, B3, D3, F2, 79, B9, D9, E9
42	5 Black Pixels	7C, BC, DC, EC, F4
43	6 Black Pixels	3F
44	6 Black Pixels	9F, CF, E7
45	6 Black Pixels	F3, F9
46	6 Black Pixels	FC

FIG. 3

```
┌─────────────────────────────────────────┐
│   EXTRACT A TILE FROM FIRST SCANLINE     │──401
└─────────────────────────────────────────┘
                    │
                    ▼
┌─────────────────────────────────────────┐
│   IDENTIFY EQUIVALENCE CLASS FOR TILE    │──402
└─────────────────────────────────────────┘
                    │
                    ▼
┌─────────────────────────────────────────┐
│     STORE EQUIVALENCE CLASS INFORMATION  │──403
│        IN FIRST SCANLINE BUFFER          │
└─────────────────────────────────────────┘
                    │
                    ▼
┌─────────────────────────────────────────┐
│      REPEAT STEPS 401-403 UNTIL ALL      │
│    THE TILES IN THE FIRST SCANLINE ARE   │──404
│     CONVERTED INTO THEIR RESPECTIVE      │
│          EQUIVALENCE CLASSES             │
└─────────────────────────────────────────┘
                    │
                    ▼
┌─────────────────────────────────────────┐
│       ENCODE EQUIVALENCE CLASSES OF      │──405
│    FIRST SCANLINE INTO LITERAL ELEMENT   │
└─────────────────────────────────────────┘
                    │
                    ▼
┌─────────────────────────────────────────┐
│    CONVERT TILES OF SECOND SCANLINE INTO │──406
│    THEIR RESPECTIVE EQUIVALENCE CLASSES  │
└─────────────────────────────────────────┘
                    │
                    ▼
┌─────────────────────────────────────────┐
│    COMPARE EQUIVALENCE CLASS ID IN THE   │──407
│ SECOND SCANLINE BUFFER TO CORRESPONDING  │
│  POSITION IN THE FIRST SCANLINE BUFFER   │
└─────────────────────────────────────────┘
```

MATCH · NO MATCH

```
┌────────────────────────────────┐   410    ┌────────────────────────────────────────┐
│  COMPARE SUCCESSIVE EQUIVALENCE │          │ ENCODE ZERO MATCH LENGTH IN COPY ELEMENT │
│ CLASS IDs IN THE RESPECTIVE     │   408    └────────────────────────────────────────┘
│ SCANLINE BUFFERS AND COMPUTE    │                            │
│ MATCH LENGTH UNTIL THERE IS NO  │                            ▼
│      LONGER A MATCH             │          ┌────────────────────────────────────────┐
└────────────────────────────────┘   411    │  COMPARE SUCCESSIVE EQUIVALENCE CLASS IDs│
              │                              │    IN THE SCANLINE BUFFER AND COMPUTE    │
              ▼                              │ DIFFERENCE LENGTH UNTIL THERE IS A MATCH │
┌────────────────────────────────┐   409    └────────────────────────────────────────┘
│ ENCODE MATCH LENGTH IN COPY     │                            │
│        ELEMENT                  │   412                       ▼
└────────────────────────────────┘          ┌────────────────────────────────────────┐
              │                              │    ENCODE LITERAL ELEMENT LENGTH AND     │
              │                              │     EQUIVALENCE CLASS IDENTIFIERS        │
              │                              └────────────────────────────────────────┘
              │                                                 │
              └──────────────────────┬──────────────────────────┘
                                     ▼
              ┌─────────────────────────────────────────┐
              │  ALTERNATE CREATION OF COPY AND LITERAL   │──413
              │    ELEMENTS FOR REMAINDER OF SCANLINE     │
              └─────────────────────────────────────────┘
```
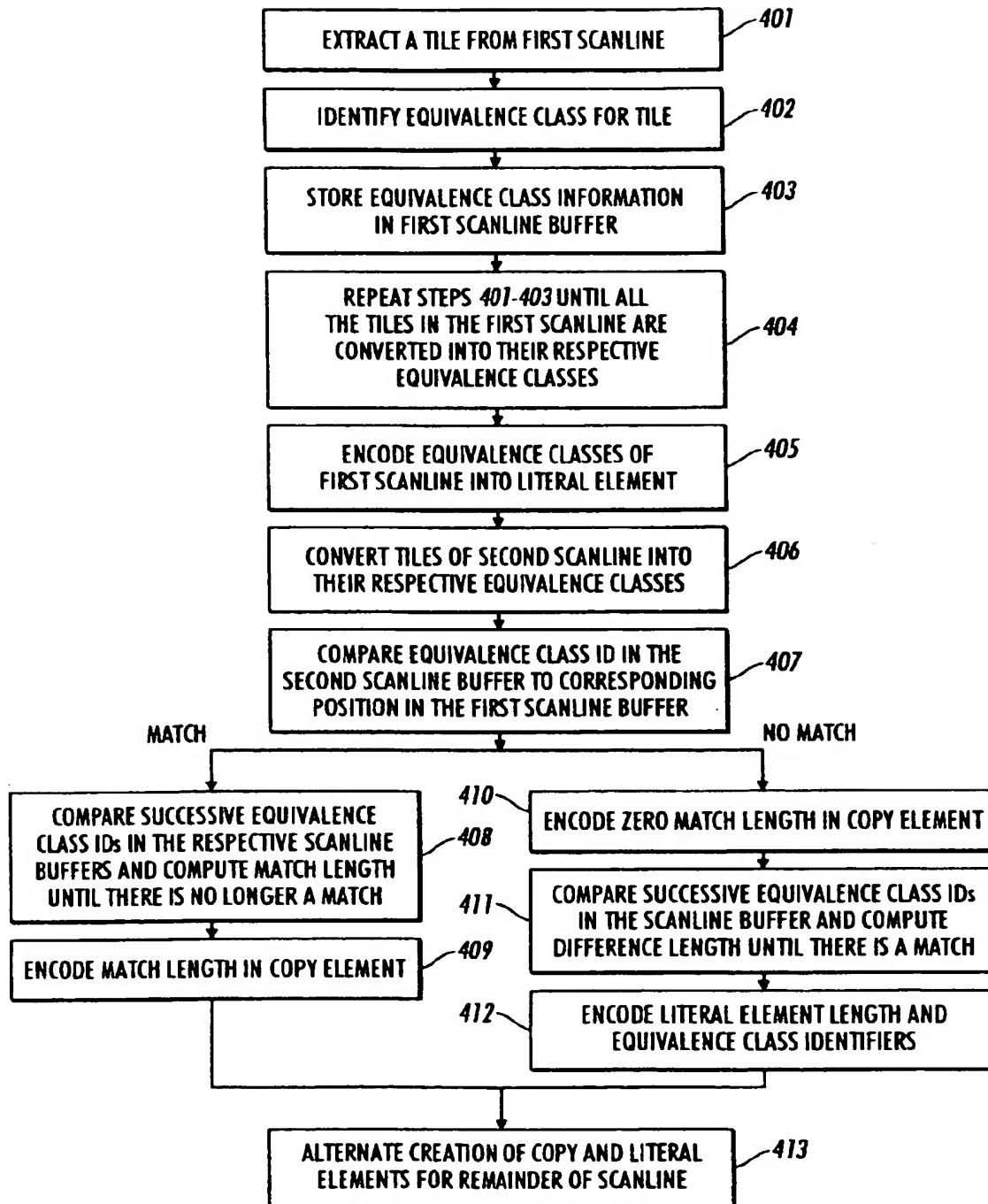
FIG. 4

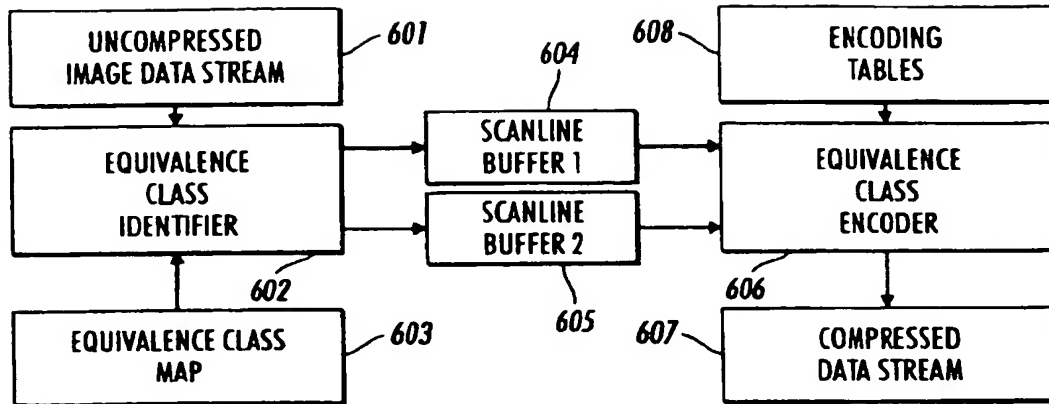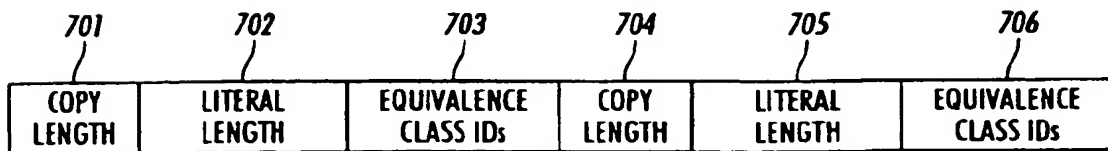|	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	4	5	5	11	5	6	10	15	5	6	6	19	10	18	14	22
1x	5	6	6	19	6	7	18	26	9	17	17	31	14	25	22	35
2x	5	6	6	19	6	7	18	26	6	7	7	30	18	29	25	39
3x	9	17	17	31	17	28	32	39	13	24	24	39	21	39	39	43
4x	5	6	6	19	6	7	18	26	6	7	7	30	18	29	25	40
5x	6	7	7	30	7	34	29	0	17	28	28	0	25	0	0	1
6x	9	17	17	31	17	28	32	40	17	28	28	0	32	0	0	1
7x	13	24	24	41	24	0	0	1	21	41	0	1	42	1	1	2
8x	5	6	6	19	6	7	18	26	6	7	7	30	18	29	25	36
9x	6	7	7	30	7	34	29	40	17	28	28	40	25	40	40	44
Ax	6	7	7	30	7	34	29	40	7	34	34	0	29	0	0	1
Bx	17	28	28	41	28	0	0	1	24	41	0	1	42	1	1	2
Cx	8	16	16	33	16	27	31	36	16	27	27	40	31	40	40	44
Dx	16	27	27	41	27	0	0	1	31	41	0	1	42	1	1	2
Ex	12	23	23	37	23	40	40	44	23	41	0	1	42	1	1	2
Fx	20	37	41	45	42	1	1	2	38	45	1	2	46	2	2	3

FIG. 5

13

FIG. 6

FIG. 7

14

DECODE FIRST SCANLINE INTO EQUIVALENCE CLASSES AND STORE IN CURRENT SCANLINE BUFFER — 801

IDENTIFY RENDERING EXEMPLARS FOR EACH EQUIVALENCE CLASS IN THE CURRENT SCANLINE BUFFER AND OUTPUT AS PART OF DECOMPRESSED BINARY IMAGE — 802

OBTAIN COPY ELEMENT FOR THE NEXT SCANLINE AND SWITCH CURRENT SCANLINE BUFFER TO REFERENCE SCANLINE BUFFER — 803

DETERMINE COPY LENGTH N FROM COPY ELEMENT — 804

COPY N EQUIVALENCE CLASSES IDs IN THE CORRESPONDING POSITIONS OF THE REFERENCE SCANLINE BUFFER INTO THE CURRENT SCANLINE BUFFER — 805

DECODE LITERAL ELEMENT TO OBTAIN LENGTH M — 806

OBTAIN M EQUIVALENCE CLASS IDs AND COPY INTO CURRENT SCANLINE BUFFER — 807

REPEAT STEPS 802-807 UNTIL HAVE ENOUGH EQUIVALENCE CLASS IDs TO FILL SCANLINE — 808

IDENTIFY RENDERING EXEMPLARS FOR EACH EQUIVALENCE CLASS OF THE CURRENT SCANLINE BUFFER AND OUTPUT AS PART OF A DECOMPRESSED BINARY IMAGE — 809

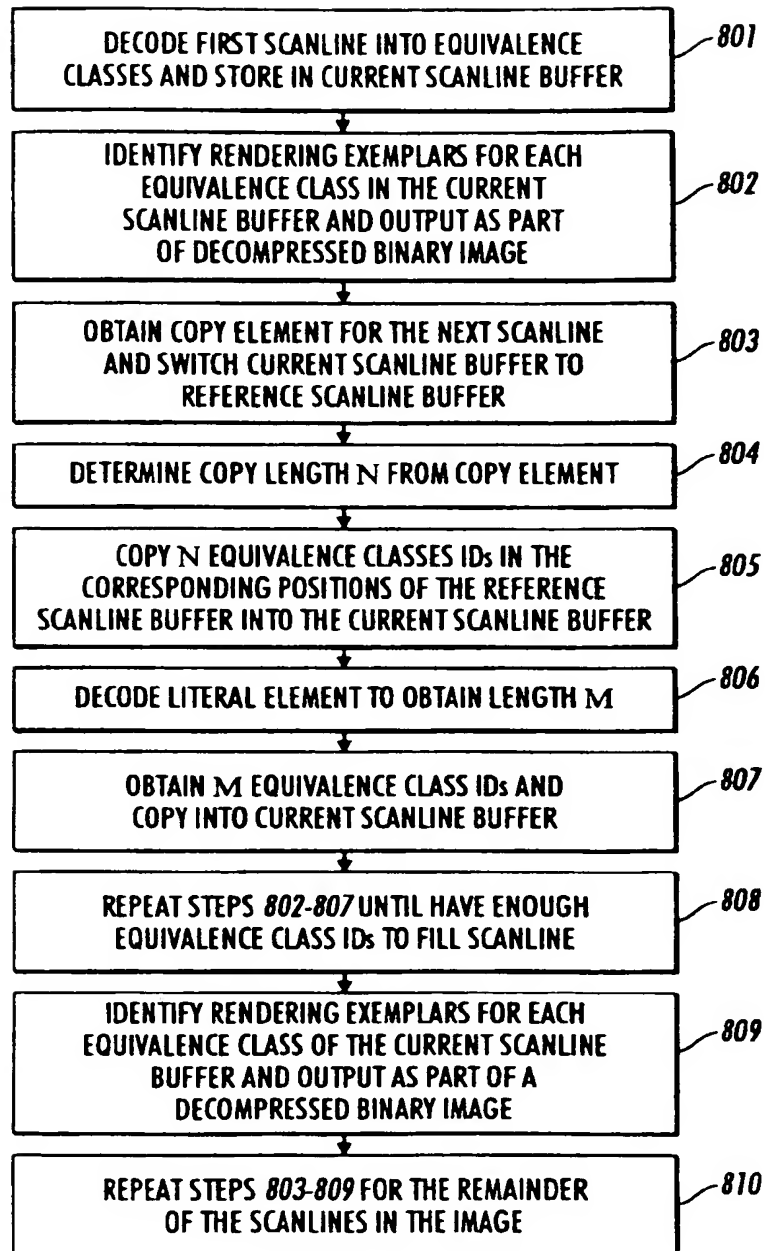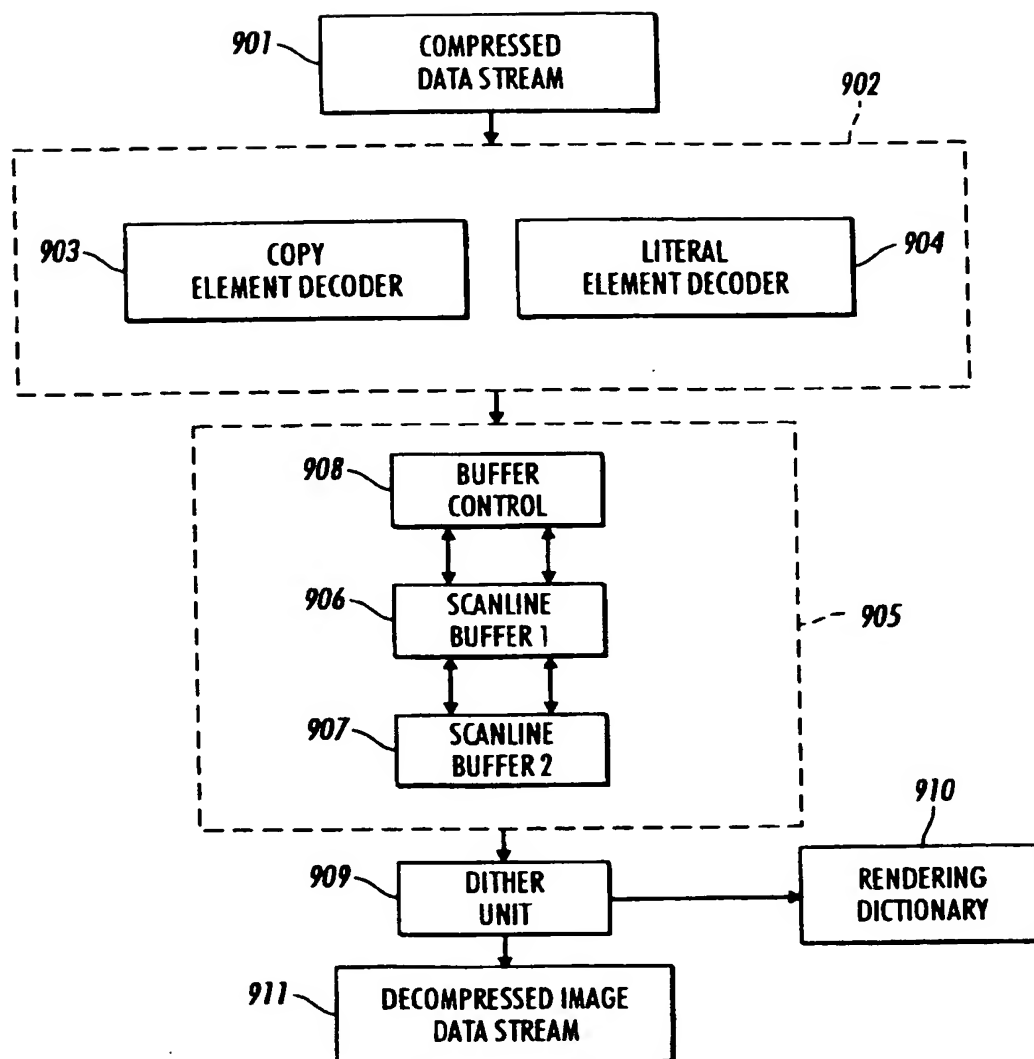REPEAT STEPS 803-809 FOR THE REMAINDER OF THE SCANLINES IN THE IMAGE — 810

FIG. 8

FIG. 9

CLASS NUMBER

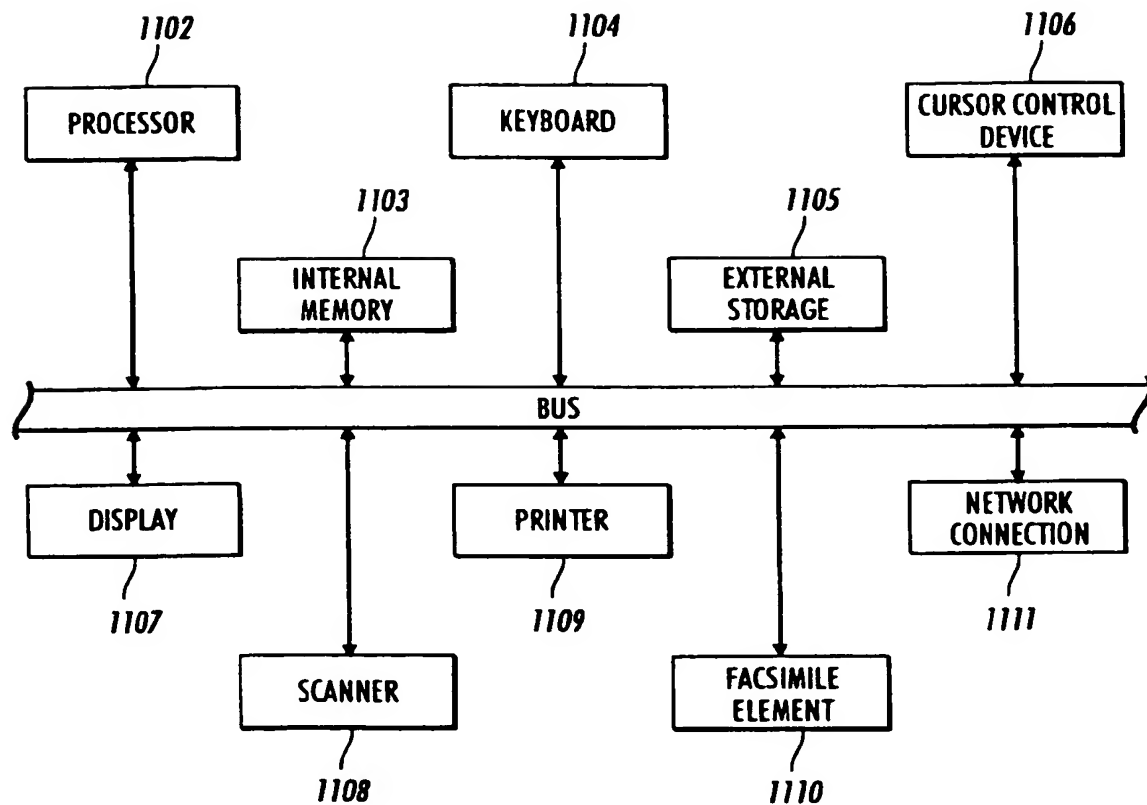|								
0	57	EA	AD	D6	6B	D5	AE	D5
1	5F	EB	BE	D7	7E	F5	BB	EE
2	EF	F7	BF	FD	7F	FB	FE	DF
3	FF	FF	FF	FF	FF	FF	FF	FF
4	00	00	00	00	00	00	00	00
5	10	08	40	02	80	04	01	20
6	A0	14	41	28	81	0A	44	11
7	A8	15	52	29	94	2A	51	2A
8	C0	C0	C0	C0	C0	C0	C0	C0
9	60	60	18	18	60	60	18	18
10	06	06	0C	0C	06	06	0C	0C
11	03	03	03	03	03	03	03	03
12	E0	E0	E0	E0	E0	E0	E0	E0
13	70	70	38	38	70	70	38	38
14	1C	1C	0E	0E	1C	1C	0E	0E
15	07	07	07	07	07	07	07	07
16	C4	D0	C2	C8	C1	C4	D0	C1
17	68	62	98	19	34	B0	1A	31
18	8C	0D	46	16	2C	8C	26	16
19	83	0B	43	13	83	0B	43	13
20	F0	F0	F0	F0	F0	F0	F0	F0
21	78	78	3C	3C	78	78	3C	3C
22	1E	1E	0F	0F	1E	1E	0F	0F
23	E8	E2	E4	E1	E8	E2	E4	E1
24	74	71	B8	3A	72	74	3A	B8
25	9C	1D	8E	2E	5C	9C	4E	8E
26	87	17	47	27	87	17	47	27
27	D4	C9	D1	CA	D2	C5	D1	CA
28	6A	65	5A	99	35	B2	59	9A
29	AC	2D	A6	56	4D	AC	56	96
30	A3	53	8B	2B	A3	53	93	4B
31	CC	CC	33	33	CC	CC	33	33
32	6C	6C	36	36	66	6C	36	36
33	C3	C3	C3	C3	C3	C3	C3	C3
34	AA	55	AA	55	AA	55	AA	55
35	1F	1F	1F	1F	1F	1F	1F	1F
36	8F	C7	8F	C7	8F	C7	8F	C7
37	E3	F1	E3	F1	E3	F1	E3	F1
38	F8	F8	F8	F8	F8	F8	F8	F8
39	3B	2F	3D	37	3E	3B	2F	3E
40	97	9D	67	E6	CB	4F	E5	CE
41	73	F2	D9	E9	D3	73	D9	E9
42	7C	F4	BC	EC	7C	F4	BC	EC
43	3F	3F	3F	3F	3F	3F	3F	3F
44	60	60	18	18	60	60	18	18
45	F3	F3	F9	F9	F3	F3	F9	F9
46	FC	FC	FC	FC	FC	FC	FC	FC

FIG. 10

1102

PROCESSOR

1104

KEYBOARD

1106

CURSOR CONTROL DEVICE

1103

INTERNAL MEMORY

1105

EXTERNAL STORAGE

BUS

DISPLAY

1107

SCANNER

1108

PRINTER

1109

FACSIMILE ELEMENT

1110

NETWORK CONNECTION

1111

FIG. 11